KKM

- $f(x)$-traverzovanie
- tokeny traverzujú/označujú územia
- levely: keď sa stretnú dva, vznikne nový
- naháňanie

```
var lev_p            : integer     init −1 ;
    cat_p, wait_p    : P           init udef ;
    last_p           : Neigh_p     init udef ;

begin if p is initiator then
        begin lev_p := lev_p + 1 ; last_p := trav(p, lev_p) ;
              cat_p := p ; send ⟨ annex, p, lev_p ⟩ to last_p
        end ;
      while ... (* Termination condition, see text *) do
       begin receive token (q, l) ;
             if token is annexing then t := A else t := C ;
             if l > lev_p then (* Case I *)
               begin lev_p := l ; cat_p := q ;
                     wait_p := udef ; last_p := trav(q, l) ;
                     send ⟨ annex, q, l ⟩ to last_p
               end
             else if l = lev_p and wait_p ≠ udef then (* Case II *)
               begin wait_p := udef ; lev_p := lev_p + 1 ;
                     last_p := trav(p, lev_p) ; cat_p := p ;
                     send ⟨ annex, p, lev_p ⟩ to last_p
               end
             else if l = lev_p and last_p = udef then (* Case III *)
               wait_p := q
             else if l = lev_p and t = A and q = cat_p then (* Case IV *)
               begin last_p := trav(q, l) ;
                     if last_p = decide
                       then p announces itself leader
                       else send ⟨ annex, q, l ⟩ to last_p
               end
             else if l = lev_p and ((t = A and
                  q > cat_p) or t = C) then (* Case V *)
               begin send ⟨ chase, q, l ⟩ to last_p ; last_p := udef end
             else if l = lev_p then (* Case VI *)
               wait_p := q
       end
end
```

počet správ

- naháňacie: 1 na vrchol a level, spolu n za level
- objavovacie: $\sum_i f(n_i)$
- ak f je konvexná, t.j. $f(a) + f(b) \leq f(a+b)$, tak je $O(\log n(n + f(n)))$ správ

K_n: vplyv orientácie

zmysel pre orientáciu

Porty sú číslované podľa vzdialenosti vo fixnej Hamiltonovej kružnici.

- algoritmus: zajať fixný pattern $\{i[1..k], i[2k], i[3k], \ldots, i[n-k]\}$
- prvá fáza $i[1..k]$
  - *level*, *ID*
  - *level* je počet zajatých
  - pripája sa celé územie (planarita!)
- druhá fáza $i[2k], i[3k], \ldots, i[n-k]$
  - nastav *owner* vrcholom $i[1..k]$, ack
  - pošli *elect* do $i[2k], i[3k], \ldots, i[n-k]$
  - *elect*: ak je v prvej fáze, alebo je slabší $\Rightarrow$ accept

počet správ

- prvá fáza: $O(n)$: z planarity
- druhá fáza: max $O(n/k)$ kandidátov, každý pošle n/k správ $\Rightarrow$ $O(n^2/k^2)$ správ

čas

- po zobudení (najsilnejšieho) $O(k)$, v najhoršom $O(n)$
- pridať wakeup fázu (poslať od $i[1], i[k]$) $\Rightarrow$ $O(k + n/k)$
- $k := \sqrt{n}$

vylepšiť čas

- prvá fáaza: zajať $i[k], i[2k], \ldots, i[n-k]$ sekvenčne
- vzniknú nezávislé "kruhy" $R_0, \ldots, R_{k-1}$, $R_j = \{i[j+k], i[j+2k], \ldots, i[j+n-k]\}$
- druhá fáza: zajať $i[1..k-1]$ v $\log k$ krokoch
  - level, ID
  - v prvom kroku zajať $i[k/2]$
  - v l-tom kroku zajať $i[k/2^l], i[3k/2^l], \ldots, i[(2^l-1)k/2^l]$ ako v úplnom grafe (za zajatých bojuje šéf)
  - v l-tom kroku je $k/2^l$ kandidátov; je $\log n$ krokov $\Rightarrow O(k \log n)$ správ
- $k := n/\log n$

vplyv synchrónnosti

algoritmy založené na porovnaniach

- ekvivalentné okolia
- c-symetrický reťazec: pre každé $\sqrt{n} \leq l \leq n$ a pre každý segment S je $\lfloor \frac{cn}{l} \rfloor$ ekvivalentných
- bit-reversal je $1/2$-symetrický
- c-symetrický reťazec, alg. nemôže skončiť po k kolách pre $\left\lfloor \frac{cn}{2k+1} \right\rfloor \geq 2$
- počet správ: $k = \left\lfloor \frac{cn-2}{4} \right\rfloor$, je aspoň $k+1$ kôl
- aspoň $\left\lfloor \frac{cn}{2r-1} \right\rfloor$ aktívnych v r-tom kole pošle správu

algoritmy využívajúce integer ID

- rôzne rýchlosti
- v i-tom kroku test

cvičenie

broadcasting a voľba šéfa na (ne?)orientovanej hyperkocke s lineárnym počtom správ